IN THE SPECIFICATION:

Please amend the paragraph beginning at page 8, line 9, as follows

Turning now to FIG. 1, a system 100 is shown for generation of executable modules for devices and for using the executable modules. System 100 comprises an automatic class generator 105, a runtime support application programmer interface (API) 150, a data manager 155, a number of devices 160, instance data 165, a client-side runtime module 170, and a user environment 180. Automatic class generator comprises a processor 106 and a memory 107. The memory 107 comprises a compiler 110 having a number of configuration classes (CCs) 115, a number of input ~~input~~ rules 120, a class builder module 125, a rule parser module 130, a source code module 135, and classes 145. It should be noted that configuration classes may be referred to herein as managed object classes (MOCs), and some of the examples shown below use the MOC acronym as part of names for configuration classes.

Please amend the paragraph beginning at page 12, line 8, as follows

Table and table entry configuration classes collectively are used to support table access. A table configuration class specifies a table entry class that forms its rows and a set of primary keys that are used to index into the table. The keys are typically configuration elements of the table entry class referenced in the table class. For example, an interface table configuration class to describe the interfaces MIB can use a table entry configuration class, say called interfaceEntry, that describes the data in each row. The table entry configuration class would typically specify ~~specifies~~ that its configuration element ifIndex, is the primary key to the table. Note that the primary key may be composed of more than one configuration element from the table entry class.

Please amend the paragraph beginning at page 16, line 3, as follows.

To illustrate how triggering occurs in an exemplary embodiment herein, a configuration element specific implicit rule is shown in FIG. 10 for the SiteData configuration class. How the rule is triggered is shown in FIG. 11. The rule in FIG. 10 accesses configuration elements serverID and serverType. Hence, the rule is deferred triggered in the set methods (see reference 1110 of FIG. 11) and direct triggered in the update methods (see reference 1120 of FIG. 11) of these configuration elements, respectively. This ensures that checks performed by this rule are always executed whenever these configuration elements are modified. Note that the triggered rule name in FIG. 11 is prefixed by the configuration element name (serverName) since this is a configuration element specific rule. This ensures that if other configuration element specific rules with the same name (iEMS_validationRule) are defined in the rules definition file, they can be triggered unequivocally. The set methods perform deferred triggering by adding the rule to a queue via a support method addToUpdateList so that they can be executed later. In contrast, the update methods perform ~~performs~~ direct triggering: the rule is directly invoked as a method call from its body. Also note that for range restricted configuration elements (e.g., serverID) a range check is also performed. The range check methods are not shown herein for clarity.

Please amend the paragraph beginning at page 20, line 9, as follows.

This section contains exemplary steps for rule detection and processing in order to create executable modules. The steps performed assumes that two passes are performed using input rules because it is generally not possible to generate all the required information and enforce rules in one pass. The order of the steps is only exemplary.

Please amend the paragraph beginning at page 22, line 18, as follows.

Step 15. For every read set configuration element of an implicit rule (e.g., for the current configuration class or for any of its configuration elements), four methods are generated: get, set, update and assign.

5